Homework Assignment #2
Due at midnight Tuesday 10/25

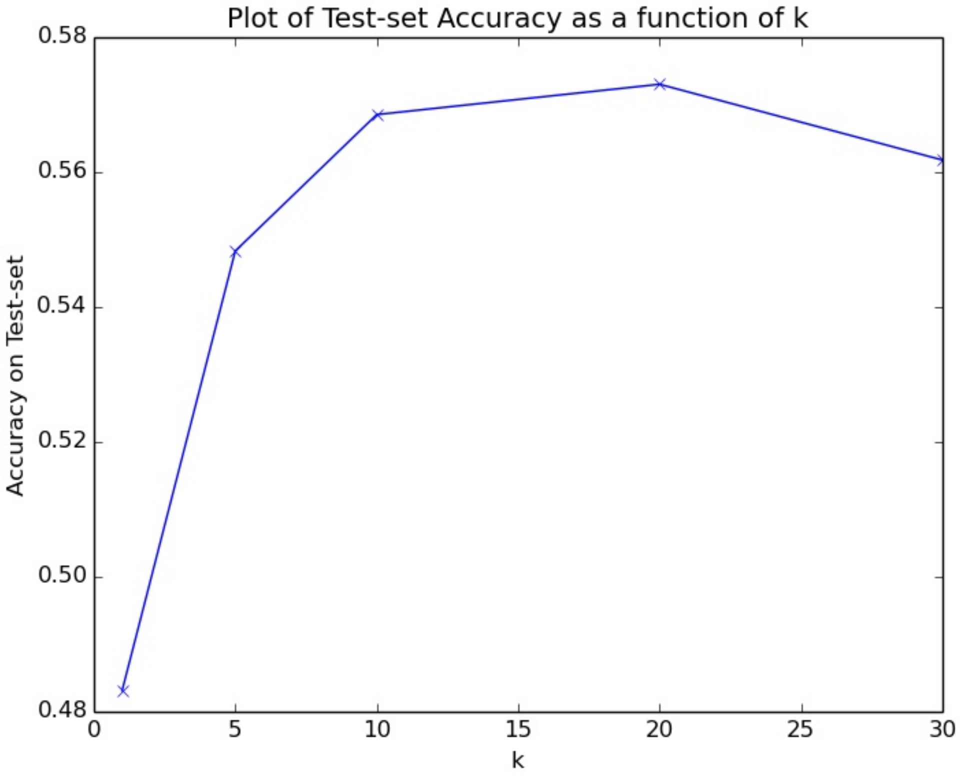
Part-1

Programming a k-nearest neighbor learner for both classification and regression.
Implementation source code files submitted. (main files: *kNN.py and kNN-select.py*)

Part-2

a) For the yeast data set, draw a plot showing how test-set accuracy varies as a function of k. Your plot should show accuracy for k = 1, 5, 10, 20, 30.

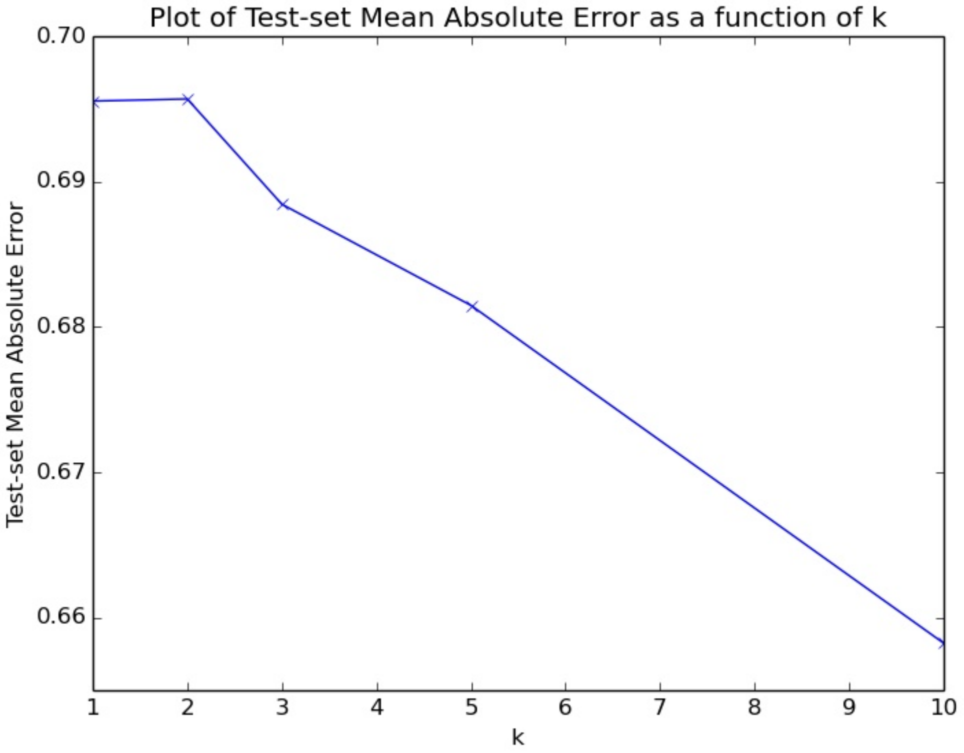
Plotting source code included in *kNN-plots.py*.



k	Predictive Accuracy
1	0.4831460674157303
5	0.5483146067415731
10	0.5685393258426966
20	0.5730337078651685
30	0.5617977528089888

b) For the wine data, draw a similar plot showing test-set mean absolute error as a function of k, for k = 1, 2, 3, 5, 10.

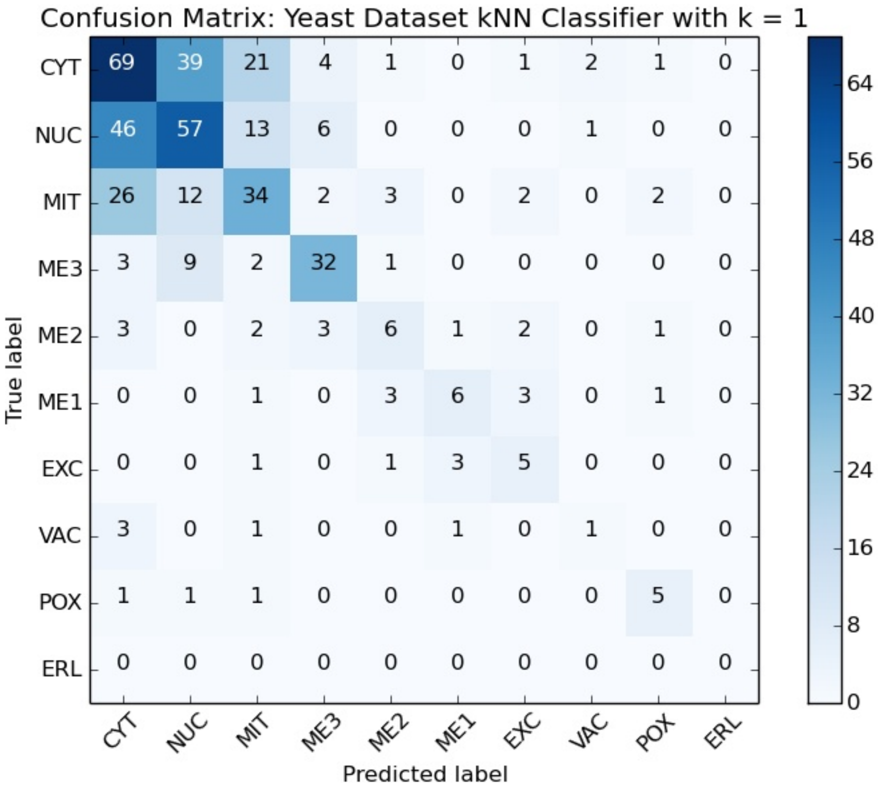
Plotting source code included in *kNN-plots.py*.

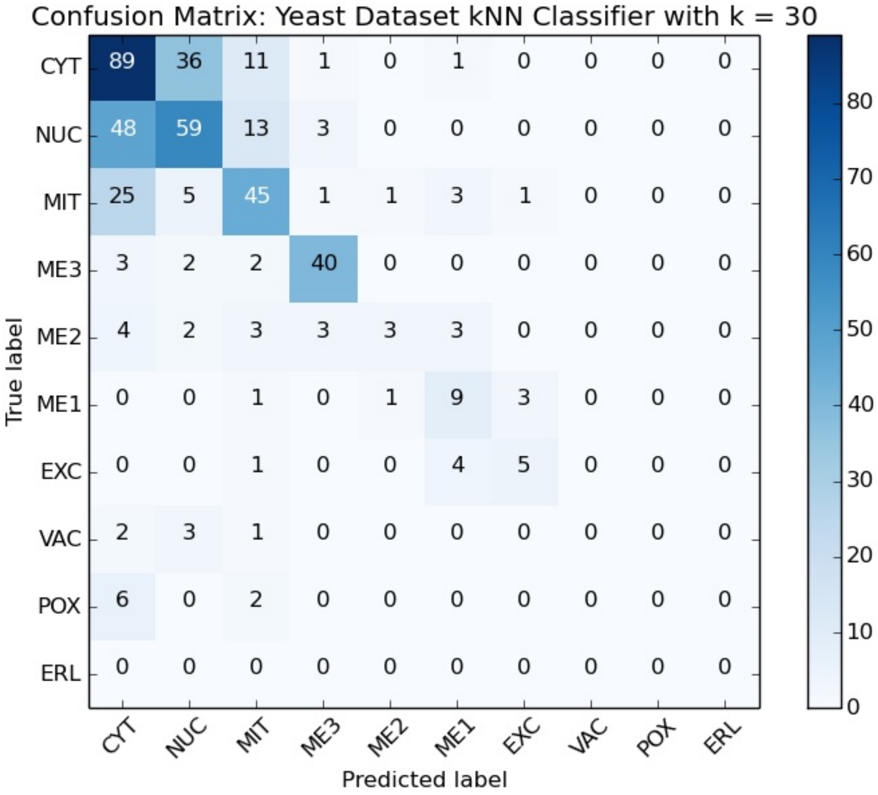


k	Mean Absolute Error
1	0.6955380577427821
2	0.6956838728492272
3	0.6884417225624573
5	0.6814814814814845
10	0.6582385535141413

c) For the yeast data set, construct confusion matrices for the k = 1 and k = 30 test-set results. Show these confusion matrices and briefly discuss what the matrices tell you about the effect of k on the misclassifications.

Confusion Matrix creation and plotting source code included in *kNN-confusionmatrix.py*.

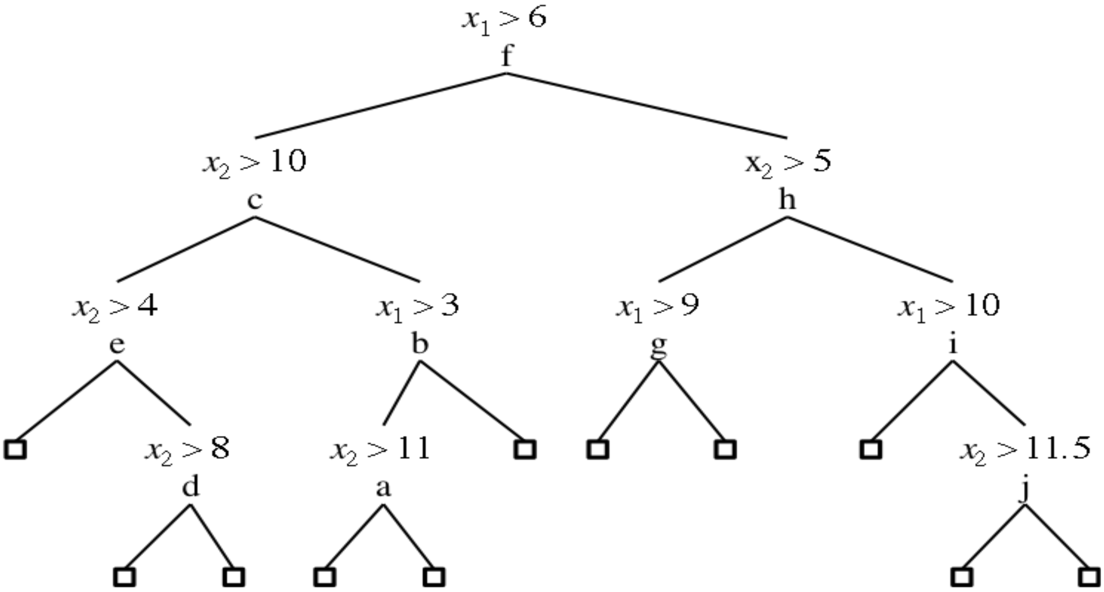




For most class labels, an increase in k from 1 to 30 has resulted in decrease (or maintenance) in number of misclassifications. For class labels that are highly represented i.e. have a significant number of instances in the training data set (like CYT, NUC, MIT, and ME3) this effect is extremely evident. However, for some classes (like ME2, VAC, and POX) the misclassifications have increased when k=30.This could be due to the small number of instances belonging to these labels in the training dataset. More and more instances are classified into one of the highly represented classes as k increases.

Part-3

Using the k-d tree displayed in the figure below, show how the nearest neighbor for x(q) = (7, 10) is found. For each step in the search, show the distance to the current node, the best distance found so far, the best node found so far, and the contents of the priority queue. Use Euclidean distance.



Instance	a	b	c	d	e	f	g	h	i	j
(x, y)	(2, 11)	(3, 12)	(5, 10)	(2, 8)	(2, 4)	(6, 3)	(9, 2)	(12, 5)	(10,10)	(13,11.5)

Query Instance : x(q) = (7, 10)

Current Node	Current Node Distance from Query Instance	Best Distance	Best Node	Priority Queue
		∞		(f, 0)
f	5√2 = 7.07106781187	5√2	f	(h, 0) (c, 1)
h	5√2 = 7.07106781187	5√2	f	(i, 0) (c, 1) (g, 5)
i	√9 = 3	3	i	(c, 1) (j, 3) (g, 5)
c	√4 = 2	2	c	(e, 0) (b, 0) (j, 3) (g, 5)
e	√61 = 7.81024967591	2	c	(b, 0) (d, 0) (j, 3) (g, 5)
b	2√5 = 4.472135955	2	c	(d, 0) (j, 3) (a, 4) (g, 5)
d	√29 = 5.38516480713	2	c	(j, 3) (a, 4) (g, 5)
j	Return c	Since (j priority = 3) > (best distance =2)		